Autoregressive PINNs for Time-Dependent PDEs

Mayank Nagda[1], Jephte Abijuru[1], Phil Ostheimer[1], Jan C. Aurich[1], Stephan Mandt[2], Marius Kloft[1], Sophie Fellenz[1]
[1]RPTU University Kaiserslautern-Landau, Germany
[2]University of California, Irvine, USA
mnagda@rptu.de

Abstract

Accurately solving time-dependent partial differential equations (PDEs) is central to many areas of science and engineering. Physics-Informed Neural Networks (PINNs) use deep learning to solve PDEs, but their pointwise predictions ignore the autoregressive nature of dynamical systems, often leading to instability and error accumulation. We propose Physics-Informed Autoregressive Networks (PIANO), a novel framework that redefines PINNs for modeling dynamical systems. PIANO predicts future states conditioned on past ones, enforcing physical consistency through self-supervised rollouts. Our theoretical analysis shows that while PINNs suffer from temporal instability, PIANO achieves stability through autoregressive modeling. Across a range of challenging time-dependent PDEs, PIANO delivers state-of-the-art accuracy and stability, and it also surpasses existing methods in weather forecasting.

1 Introduction

In 1814, Laplace noted that *"the present state of a system is the effect of its past and the cause of its future"* [De Laplace, 1995], capturing a key principle of dynamics: future states unfold autoregressively from the present. From weather prediction to heat diffusion and fluid flow, governed by time-dependent PDEs, learning hinges on how the current state shapes the next. Yet popular ML solvers such as PINNs [Raissi et al., 2019] predict each time step independently, without conditioning on prior states, which we show can induce temporal instability and compounding errors.

Autoregressive (AR) models compute the state $u(\cdot, t_n)$ from past states, e.g., $u(\cdot, t_n) = f(u(\cdot, t_{n-1}), u(\cdot, t_{n-2}), \dots)$, whereas non-AR models estimate each $u(\cdot, t_n)$ directly from coordinates, $u(\cdot, t_n) = f(\cdot, t_n)$. PINNs enforce PDE residuals but are non-AR: they map $(\cdot, t) \mapsto u(\cdot, t)$ pointwise. While effective in domains like fluid mechanics [Cai et al., 2021] and cardiovascular flows [Raissi et al., 2020], they often struggle on dynamical accuracy [Wang et al., 2024]. Recent variants (PINNsFormer [Zhao et al., 2024], PINNMamba [Xu et al., 2025]) introduce sequence-aware encoders over $(\cdot, t)$ yet still predict $u(\cdot, t_n)$ independently of $u(\cdot, t_{n-1})$, remaining non-AR.

We show that standard PINNs exhibit temporal instability with errors growing over time. To this end, we introduce PIANO, which enforces physics while predicting each $u(\cdot, t_n)$ conditioned on prior states to curb error growth. As illustrated in Fig. 1, PIANO learns a state transition that propagates solutions reliably from initial conditions. Our contributions are: (i) theory: PINNs are temporally unstable for time-dependent PDEs, while AR modeling yields stability (Sec. 2); (ii) method: a physics-informed AR framework that reformulates PINNs to model temporal evolution autoregressively (Sec. 2); (iii) empirics: on challenging PDEs and weather forecasting, PIANO achieves better accuracy and stability over existing methods (Sec. 3).

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025

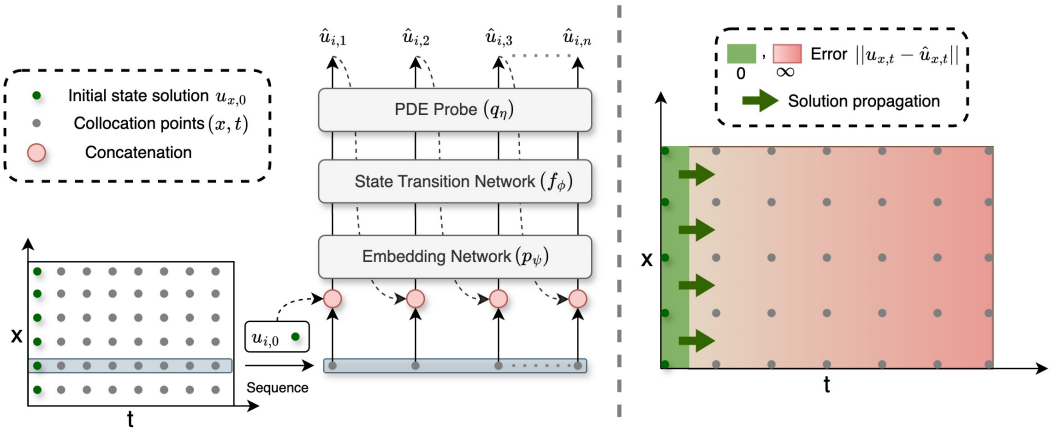


Figure 1: PIANO overview. For each fixed x_i, inputs $\{(x_i, t_j)\}_{j=0}^{n}$ are concatenated with the previous prediction $\hat{u}(x_i, t_{j-1})$ and passed through an embedding network p_ψ, state transition network f_ϕ, and PDE probe q_η. Rollouts start from the known initial condition $u(x, t_0)$.

2 Method

Setup and limitation of standard PINNs. We consider time-dependent PDEs on a domain $\Omega \subset \mathbb{R}^d$ with solution $u : \mathbb{R}^d \to \mathbb{R}^l$. Interior, initial, and boundary constraints are encoded by operators $\mathcal{O}_\Omega, \mathcal{O}_{\Omega_0}, \mathcal{O}_{\partial\Omega}$ (e.g., for heat, $\mathcal{O}_\Omega(u) = u_t - u_{xx}$). A PINN [Raissi et al., 2019] approximates u with u_θ by minimizing a residual loss $\mathcal{L}(u_\theta) = \sum_{X \in \{\Omega, \Omega_0, \partial\Omega\}} \frac{\lambda_X}{N_X} \sum_{i=1}^{N_X} \|\mathcal{O}_X(u_\theta)(x_X^{(i)})\|^2$. However, standard PINNs are non-AR: they predict $u(\cdot, t_n)$ directly from $(\cdot, t_n)$, independent of $u(\cdot, t_{n-1})$, which can cause temporal drift.

We formalize this drift via the true evolution operator $\mathcal{G}(\Delta t)$, which maps the state of the system at time t_n to the state at time $t_{n+1} = t_n + \Delta t$. Defining the error $e_n(\cdot) = u_\theta(\cdot, t_n) - u_{\text{true}}(\cdot, t_n)$ and the one-step rollout error $\delta_n = \|u_\theta(\cdot, t_{n+1}) - \mathcal{G}(\Delta t)[u_\theta(\cdot, t_n)]\|_2$, we show the recursion $\|e_{n+1}\|_2 \le L_{\mathcal{G}} \|e_n\|_2 + \delta_n$. Because the usual residual loss does not constrain δ_n, fresh error can be injected at each step, accumulating over time. Full statements and proofs are in Appendix A.

PIANO Architecture. To achieve temporal consistency, PIANO conditions each prediction on the previous state. For each spatial location x_i, we process a temporal sequence $\{(x_i, t_j)\}_{j=0}^{M}$. At step t_j, the model input is $((x_i, t_j), \hat{u}(x_i, t_{j-1}))$, i.e., the current coordinate concatenated with the previous prediction. The architecture (Figure 1) has three components:

- *Embedding network* p_ψ: maps the input tuple to a feature vector m_j^i, enriching spatial–temporal context and the carried-over state.
- *State-transition network* f_ϕ: a lightweight state-space module that maintains a latent state h and updates it autoregressively, $h_j^i = \sigma(\text{LN}(\mathbf{A}h_{j-1}^i + \mathbf{B}m_j^i))$, producing an output representation $o_j^i = \mathbf{C}h_j^i + \mathbf{D}m_j^i + m_j^i$. This stage models the *evolution* from t_{j-1} to t_j.
- *PDE probe* q_η: decodes o_j^i to the physical field, yielding $\hat{u}(x_i, t_j) = q_\eta(o_j^i)$.

Rollouts start from the known initial condition $u(x, t_0)$, which anchors trajectories and prevents arbitrary offsets.

Physics-Informed Experience Learning (PIEL). We train by unrolling the model in time and enforcing physics along the entire predicted trajectory. Starting from $u(x, t_0)$, we generate $\hat{u}(x_i, t_j) = u_\theta(x_i, t_j, \hat{u}(x_i, t_{j-1}))$ for $j = 1, \ldots, M$, backpropagating through time. PDE derivatives are computed on the predicted space–time grid using second-order finite differences. Over $X \in \{\Omega, \partial\Omega\}$, we accumulate residual energy $\mathcal{E}_X(x_i) = \frac{1}{M} \sum_{j=1}^{M} \|\mathcal{O}_X[\hat{u}](x_i, t_j)\|^2$ and define the objective $\mathcal{L}_{\text{PIANO}} = \sum_X \frac{\lambda_X}{N_X} \sum_{i=1}^{N_X} \mathcal{E}_X(x_i)$. When data are available, we add a simple teacher-forcing term that replaces $\hat{u}(x_i, t_{j-1})$ with the true state during training; inference always uses free-rollout (the model feeds on its own predictions). Training algorithm is provided in App. B.

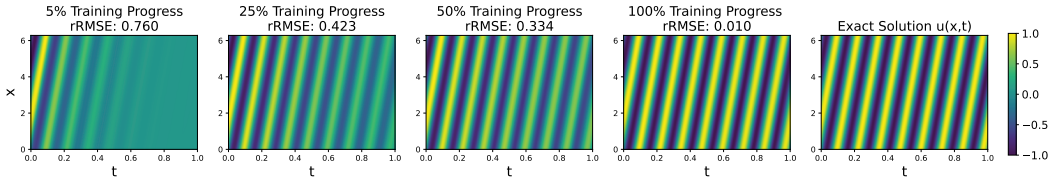
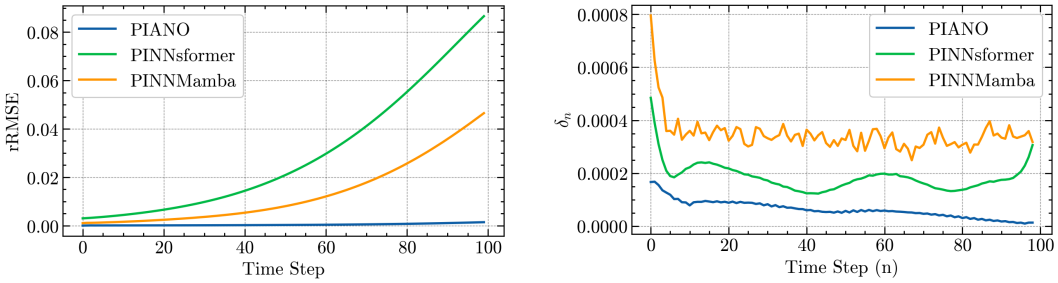


Figure 2: Training dynamics of PIANO on the convection equation. At 5%, predictions remain accurate only around the initial condition; by 25–50%, temporal propagation improves noticeably, and at convergence the predictions become visually indistinguishable from the exact solution. This demonstrates that propagating accurate information from the initial state leads to stable convergence without intermediate failures.



(a): Rollout error (rRMSE) vs. time.

(b): One-step defect δ_n vs. time.

Figure 3: PIANO exhibits almost constant error growth and achieves the lowest one-step defect, whereas PINNsFormer and PINNMamba show rapidly increasing rollout errors and larger δ_n. In both panels, lower values indicate better performance.

Theoretical guaranties for PIANO. The autoregressive design directly targets the transition error. Under a p-th order time and q-th order space discretization, our main result (Appendix A) gives $\delta_n \leq \rho + \kappa(\Delta t^p + h^q)$, where ρ is the residual energy minimized by $\mathcal{L}_{\text{PIANO}}$ and $\kappa > 0$ depends on the discretization; for the common second-order case ($p = q = 2$), mesh refinement reduces the consistency term, while minimizing ρ suppresses δ_n. Intuitively, PIANO learns a stable evolution map: each step is both *physically consistent* and *conditioned on the past*, curbing error injection and accumulation over long horizons.

3 Experiments

To empirically demonstrate the effectiveness of PIANO, we evaluate its performance on PDE benchmarks (Section 3.1) and a real-world weather forecasting task (Section 3.2).

3.1 PDE Benchmarks

Benchmark We evaluate PIANO on four canonical time-dependent PDEs—Wave, Reaction, Convection, and Heat—spanning higher-order dynamics, nonlinearity, transport, and stiffness. Baselines include MLP-PINNs [Raissi et al., 2019], FLS [Wong et al., 2022], QRes [Bu and Karpatne, 2021], KANs [Liu et al., 2025], RoPINNs [Wu et al., 2024], and sequential but non-AR models PINNsFormer [Zhao et al., 2024] and PINNMamba [Xu et al., 2025]. All methods use comparable samples and training budget; PIANO employs a state-space architecture trained on a 200×200 grid with AdamW. Using rMAE and rRMSE, PIANO consistently outperforms all baselines across all benchmarks, with promotion of at least 50% and at most 100%; tables and per-task breakdowns, together with complete detail and hyperparameter guidance appear in Appendix C.

Training Dynamics Figure 2 illustrates PIANO's training process on the convection equation. Beginning from the initial condition, early predictions remain localized (5%), then progressively extend forward while appearing diffuse (25–50%), and finally, at convergence (100%), become visually indistinguishable from the ground truth (rRMSE 0.010). These observations support the

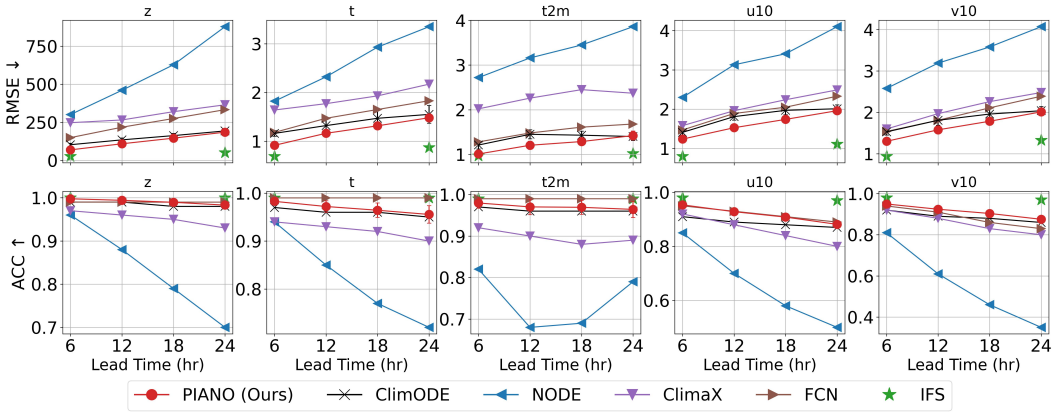


Figure 4: We compare global weather forecasting results on the ERA5 dataset, evaluating PIANO against recent baseline models. The figure presents the latitude-weighted root mean square error (RMSE) and anomaly correlation coefficient (ACC) across lead times ranging from 6 to 24 hours. The forecasting task targets five essential atmospheric variables: geopotential (`z`), air temperature (`t`), 2-meter surface temperature (`t2m`), and 10-meter wind components (`u10` and `v10`). PIANO generally delivers consistently lower errors and higher correlations, demonstrating the strength of its autoregressive, physics-informed design in capturing the dynamics of complex weather systems.

intuition that accurately propagating information from the initial state promotes stable convergence and prevents intermediate breakdowns during learning.

Temporal Stability Diagnostics We conduct stability diagnostics on the Reaction equation to empirically support our theoretical claims, with results summarized in Figure 3. Panel (a) illustrates the rollout error (rRMSE) over time, where baseline models exhibit uncontrolled error escalation, leading to rapid growth as the rollout progresses. In contrast, PIANO maintains almost constant error levels, demonstrating strong temporal stability. Panel (b) presents the one-step defect δ_n, defined in Def. A.2. By directly penalizing this defect during training, PIANO achieves consistently lower δ_n values, whereas baseline methods accumulate increasingly larger defects. These findings confirm our theoretical predictions: non-autoregressive PINNs display intrinsic instability and exponential error amplification, while PIANO's autoregressive formulation provides robust stability and reliable long-term accuracy.

3.2 Global Weather Forecasting

We forecast five ERA5 variables—atmospheric temperature (`t`), surface temperature (`t2m`), 10 m winds (`u10`, `v10`), and geopotential height (`z`)—using 6-hourly data at 5.625° resolution. Building on physics-informed formulations (e.g., ClimODE), we couple an advection term that enforces transport with a learnable updater f_θ; unlike prior work, PIANO trains autoregressively with teacher forcing (conditioning each step on the previous true state) and performs free-rollout at inference. We compare against NODE, FCN/FourCastNet, ClimaX, and ClimODE, with IFS as a numerical gold standard, and report latitude-weighted RMSE and ACC. Figure 4 summarizes the results. Across 6–24 h lead times and all variables, PIANO mostly yields lower RMSE and higher ACC, with the largest gains at shorter leads where AR conditioning curbs early error growth. Complete setup, metrics, and tables appear in App. D.2.

4 Conclusion

We present PIANO, a physics-informed autoregressive framework for solving time-dependent PDEs. Unlike traditional PINNs that accumulate instability over time, PIANO aligns with the autoregressive behavior of dynamical systems, ensuring stable and accurate predictions. Experiments on challenging PDEs and global weather forecasting show improved results, demonstrating that respecting temporal dynamics is crucial for advancing physics-informed learning in dynamical systems.

Acknowledgments

The main part of this work was conducted within the DFG Research Unit FOR 5359 (ID 459419731) on Deep Learning on Sparse Chemical Process Data. SF and MK further acknowledge support by the DFG through TRR 375 (ID 511263698), SPP 2298 (ID 441826958), and SPP 2331 (441958259), by the Carl-Zeiss Foundation through the initiatives AI-Care and Process Engineering 4.0, and by the BMFTR award 01IS24071A.

References

Jie Bu and Anuj Karpatne. Quadratic residual networks: A new class of neural networks for solving forward and inverse problems in physics involving pdes. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 675–683. SIAM, 2021.

John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12): 1727–1738, 2021.

Marquis De Laplace. *A philosophical essay on probabilities*. Courier Corporation, 1995.

ECMWF. *IFS Documentation CY48R1 - Part I: Observations*. Number 1. ECMWF, 06/2023 2023. doi: 10.21957/0f360ba4ca.

Arieh Iserles. *A first course in the numerical analysis of differential equations*, volume 44. Cambridge university press, 2009.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=Ozo7qJ5vZi`.

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Amnon Pazy. *Semigroups of linear operators and applications to partial differential equations*, volume 44. Springer Science & Business Media, 2012.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Stephan Rasp, Peter D Dueben, Sebastian Scher, Jonathan A Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020.

Yogesh Verma, Markus Heinonen, and Vikas Garg. ClimODE: Climate forecasting with physics-informed neural ODEs. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=xuY33XhEGR`.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421: 116813, 2024.

Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, and Yew-Soon Ong. Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, 5(3): 985–1000, 2022.

Haixu Wu, Huakun Luo, Yuezhou Ma, Jianmin Wang, and Mingsheng Long. Ropinn: Region optimized physics-informed neural networks. In *Advances in Neural Information Processing Systems*, 2024.

Chenhui Xu, Dancheng Liu, Yuting Hu, Jiajie Li, Ruiyang Qin, Qingxiao Zheng, and Jinjun Xiong. Sub-sequential physics-informed learning with state space model. *arXiv preprint arXiv:2502.00318*, 2025.

Zhiyuan Zhao, Xueying Ding, and B. Aditya Prakash. PINNsformer: A transformer-based framework for physics-informed neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=DO2WFXU1Be`.

A Proofs and Extended Theoretical Analysis

The standard PINN formulation often fails to produce accurate solutions for time-dependent PDEs [Zhao et al., 2024, Xu et al., 2025]. We argue that this is not simply an optimization issue, but a deeper architectural mismatch. Classical time-stepping schemes like finite difference or Runge–Kutta are explicitly autoregressive: they update the solution at time t_{n+1} using the known state at t_n, preserving how dynamical systems evolve in time [Iserles, 2009, Butcher, 2016]. In contrast, PINNs predict each state directly from coordinates $(\cdot, t)$ without conditioning on prior predictions, effectively breaking this autoregressive structure. Viewed through the lens of semigroup theory [Pazy, 2012], time-stepping methods approximate an evolution operator that advances the system forward, an operator that PINNs fail to represent. We now formalize this mismatch by defining the evolution operator.

Definition A.1 (Evolution Operator)**.** A time-dependent PDE of the form $\frac{\partial u}{\partial t} = \mathcal{F}(u, t)$ defines a dynamical system. Its solution can be described by an evolution operator, $\mathcal{G}(\Delta t)$, which maps the state of the system at time t_n to the state at time $t_{n+1} = t_n + \Delta t$:

$$u_{true}(x, t_{n+1}) = \mathcal{G}(\Delta t)[u_{true}(x, t_n)]. \tag{1}$$

A stable solver must ensure that errors do not amplify uncontrollably as this operator is applied repeatedly. Let the error of a model u_θ at time step t_n be $e_n(x) = u_\theta(x, t_n) - u_{true}(x, t_n)$. We can now define the source of instability in non-AR models.

Definition A.2 (One-Step Rollout Error)**.** Given a model's solution $u_\theta(x, t_n)$, the true physical evolution would yield the state $\mathcal{G}(\Delta t)[u_\theta(x, t_n)]$ at time t_{n+1}. The one-step rollout error is the discrepancy between the model's actual prediction at t_{n+1} and the physically evolved state:

$$\delta_n = \left\| u_\theta(x, t_{n+1}) - \mathcal{G}(\Delta t)[u_\theta(x, t_n)] \right\|_2 . \tag{2}$$

This error quantifies how poorly the model approximates the true one-step dynamics when initialized from its own prediction at the previous time step.

A.1 Proofs

Theorem A.3 (Error Propagation in PINNs)**.** *For non-autoregressive PINNs, the error at step t_{n+1} is bounded by the sum of the propagated error from the previous step t_n and the one-step rollout error δ_n:*

$$\|e_{n+1}\|_2 \leq L_{\mathcal{G}} \cdot \|e_n\|_2 + \delta_n, \tag{3}$$

where $L_{\mathcal{G}}$ is the Lipschitz constant of the true evolution operator $\mathcal{G}(\Delta t)$.

Proof. The proof relies on the key assumption that the true evolution operator, $\mathcal{G}(\Delta t)$, is Lipschitz continuous with a constant $L_{\mathcal{G}}$. This property ensures that the operator does not excessively amplify differences between input states, and it is formally stated as:

$$\|\mathcal{G}(\Delta t)[a] - \mathcal{G}(\Delta t)[b]\|_2 \leq L_{\mathcal{G}} \cdot \|a - b\|_2 \tag{4}$$

for any two system states a and b.

We begin with the definition of the error at time step t_{n+1}:

$$e_{n+1} = u_\theta(x, t_{n+1}) - u_{true}(x, t_{n+1}). \tag{5}$$

By definition, the true solution at t_{n+1} is given by the evolution operator $\mathcal{G}(\Delta t)$ applied to the true solution at t_n. Substituting this into our error expression gives:

$$e_{n+1} = u_\theta(x, t_{n+1}) - \mathcal{G}(\Delta t)[u_{true}(x, t_n)]. \tag{6}$$

We now add and subtract the term $\mathcal{G}(\Delta t)[u_\theta(x, t_n)]$. This allows us to connect the model's prediction at t_{n+1} to the evolution of its own prediction from t_n:

$$\begin{aligned} e_{n+1} = &(u_\theta(x, t_{n+1}) - \mathcal{G}(\Delta t)[u_\theta(x, t_n)]) \\ &+ (\mathcal{G}(\Delta t)[u_\theta(x, t_n)] - \mathcal{G}(\Delta t)[u_{true}(x, t_n)]) . \end{aligned} \tag{7}$$

Taking the L_2 norm and applying the triangle inequality ($\|A + B\| \leq \|A\| + \|B\|$) yields:

$$\begin{aligned} \|e_{n+1}\|_2 \leq &\|u_\theta(x, t_{n+1}) - \mathcal{G}(\Delta t)[u_\theta(x, t_n)]\|_2 \\ &+ \|\mathcal{G}(\Delta t)[u_\theta(x, t_n)] - \mathcal{G}(\Delta t)[u_{true}(x, t_n)]\|_2 . \end{aligned} \tag{8}$$

We recognize the first term on the right-hand side as the definition of the one-step rollout error, δ_n. For the second term, we apply the Lipschitz continuity of the operator $\mathcal{G}$:

$$\begin{aligned} &\|\mathcal{G}(\Delta t)[u_\theta(x, t_n)] - \mathcal{G}(\Delta t)[u_{true}(x, t_n)]\|_2 \\ &\quad \leq L_\mathcal{G} \cdot \|u_\theta(x, t_n) - u_{true}(x, t_n)\|_2 \\ &\quad = L_\mathcal{G} \cdot \|e_n\|_2. \end{aligned} \tag{9}$$

Substituting these two results back into Equation (8), we arrive at the final inequality:

$$\|e_{n+1}\|_2 \leq \delta_n + L_\mathcal{G} \cdot \|e_n\|_2. \tag{10}$$

Rearranging the terms gives the statement of the theorem:

$$\|e_{n+1}\|_2 \leq L_\mathcal{G} \cdot \|e_n\|_2 + \delta_n. \tag{11}$$

□

Theorem A.4 (Bound on One-Step Rollout Error in PIANO)**.** *Let $G(\Delta t)$ be the exact evolution operator and let $G_{h,\Delta t}$ be a finite-difference (FD) approximation of temporal order p and spatial order q, with consistency constant $\kappa > 0$. Let L_{PIANO} be the PIEL loss in Eq. (6) and define*

$$\rho \;:=\; C_{\mathrm{disc}}(\Delta t, h)\, L_{\mathrm{PIANO}}^{1/2},$$

where $C_{\mathrm{disc}}(\Delta t, h)$ collects the FD stencil constants and converts residual units to the state norm used below. Then for the one-step rollout error $\delta_n := \|\hat{u}(\cdot, t_{n+1}) - G(\Delta t)[\hat{u}(\cdot, t_n)]\|$ (Def. 3.2), we have for all $n = 0, \ldots, M-1$,

$$\delta_n \;\leq\; \rho \;+\; \kappa\big(\Delta t^{\,p} + h^{\,q}\big).$$

Proof. Fix a step n and abbreviate $\hat{u}^n := \hat{u}(\cdot, t_n)$. By adding and subtracting $G_{h,\Delta t}(\hat{u}^n)$,

$$\delta_n \;=\; \big\|\hat{u}^{n+1} - G(\Delta t)[\hat{u}^n]\big\| \;\leq\; \underbrace{\big\|\hat{u}^{n+1} - G_{h,\Delta t}(\hat{u}^n)\big\|}_{\text{(I)}} \;+\; \underbrace{\big\|G_{h,\Delta t}(\hat{u}^n) - G(\Delta t)[\hat{u}^n]\big\|}_{\text{(II)}}. \tag{12}$$

Let $\mathcal{R}_{h,\Delta t}[\hat{u}](x_i, t_j)$ denote the FD residual of the governing PDE evaluated on the predicted rollout (Eq. (5)), i.e.,

$$\mathcal{R}_{h,\Delta t}[\hat{u}](x_i, t_j) \;=\; D_t \hat{u}(x_i, t_j) - \mathcal{F}_h\big(\hat{u}(\cdot, t_j)\big)\Big|_{x_i},$$

where D_t is the chosen temporal difference operator and $\mathcal{F}_h$ the spatial FD discretization of $\mathcal{F}$. For a one-step scheme, the FD update can be written as

$$G_{h,\Delta t}(\hat{u}^n) \;=\; \hat{u}^n + \Delta t\, \mathcal{A}_{h,\Delta t}(\hat{u}^n),$$

for some (possibly nonlinear) discrete operator $\mathcal{A}_{h,\Delta t}$ induced by the stencil. A standard algebraic manipulation of the stencil (equivalently, a discrete variation-of-constants identity) yields a bounded linear map $\mathcal{B}_{h,\Delta t}$ such that

$$\hat{u}^{n+1} - G_{h,\Delta t}(\hat{u}^n) \;=\; \mathcal{B}_{h,\Delta t}(\mathcal{R}_{h,\Delta t}[\hat{u}](\cdot, t_{n+\theta}))\,, \tag{13}$$

for some $\theta \in \{0, 1/2, 1\}$ depending on the time stencil (forward/Crank–Nicolson/backward). On the discrete grid, all norms are equivalent and $\|\mathcal{B}_{h,\Delta t}\| \le C_{\mathrm{disc}}(\Delta t, h)$ for a constant that depends only on the stencil and $(\Delta t, h)$. Taking norms in Eq.13 and averaging over spatial points $\{x_i\}_{i=1}^{N_\Omega}$ gives

$$\text{(I)} \;\le\; C_{\mathrm{disc}}(\Delta t, h) \left(\frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} \left\| \mathcal{R}_{h,\Delta t}[\hat{u}](x_i, t_{n+\theta}) \right\|^2 \right)^{1/2}.$$

By definition of the residual energies and the training objective (Eq. (5)–(6)), the bracketed quantity is controlled by L_{PIANO} up to the weights λ_X/N_X and the boundary terms. Thus,

$$\text{(I)} \;\le\; C_{\mathrm{disc}}(\Delta t, h)\, L_{\mathrm{PIANO}}^{1/2} \;=\; \rho.$$

Units/Scaling. Since $\mathcal{R}_{h,\Delta t}$ has the units of the PDE operator, C_{disc} carries the complementary units to yield the state norm; therefore ρ has the units of u. Because L_{PIANO} aggregates $\frac{\lambda_X}{N_X}\sum_i E_X(x_i, \hat{u})$, the dependence on λ_X, N_X enters only as $L_{\mathrm{PIANO}}^{1/2} \propto \sqrt{\lambda_X/N_X}$.

By the (p, q)-order consistency of $G_{h,\Delta t}$ with $G(\Delta t)$ applied to the same input state $\hat{u}^n$, there exists $\kappa > 0$ such that

$$\text{(II)} \;\le\; \kappa\big(\Delta t^{\,p} + h^{\,q}\big).$$

Substituting the bounds for (I) and (II) into Eq.12 gives

$$\delta_n \;\le\; \rho \;+\; \kappa\big(\Delta t^{\,p} + h^{\,q}\big),$$

as claimed. □

B Training Algorithm

Algorithm 1 Training PIANO via Experience Learning

Initialize model parameters ψ, ϕ, η.
for each training iteration **do**
  Sample a batch of spatial coordinates $\{x_i\}_{i=1}^{N_x} \subset \Omega \cup \partial\Omega$.
  Set initial state from the known condition: $\hat{u}(x_i, t_0) \leftarrow u(x_i, t_0)$ for all i.
  Initialize hidden state $h_0^i \leftarrow \mathbf{0}$ for all i.
  for each time step t_j, for $j = 1, \ldots, M$ **do**
    Form input vector: $s_j^i = (x_i, t_j, \hat{u}(x_i, t_{j-1}))$.
    Compute embedding: $m_j^i = p_\psi(s_j^i)$.
    Update hidden state and output representation: $(h_j^i, o_j^i) = f_\phi(h_{j-1}^i, m_j^i)$.
    Predict solution: $\hat{u}(x_i, t_j) = q_\eta(o_j^i)$.
    Compute loss.
  end for
  Normalize loss over time steps and batch size.
  Update parameters ψ, ϕ, η.
end for

This training procedure, which we refer to as Physics-Informed Experience Learning (PIEL), optimizes the model to generate physically consistent solution trajectories based on its own predictions. The experience learning component comes from the autoregressive rollout described in Algorithm 1: for each spatial coordinate x_i in a batch, the model generates an entire temporal trajectory starting from the known initial condition $u(x_i, t_0)$. Each subsequent prediction $\hat{u}(x_i, t_j)$ is conditioned on the model's own previous output $\hat{u}(x_i, t_{j-1})$, which forces the model to learn from its own generated experience.

Table 1: PIANO as a robust and accurate PDE solver across a range of PDE benchmarks. rMAE and rRMSE are reported separately for each PDE. Best values are highlighted in **bold** and the second best are underlined. PIANO outperforms baselines across all benchmarks. Promotion refers to the relative error reduced w.r.t. the second best model $(1 - \frac{\texttt{Our Error}}{\texttt{Second Best Error}})$.

Model	Wave		Reaction		Convection		Heat	
	rMAE	rRMSE	rMAE	rRMSE	rMAE	rRMSE	rMAE	rRMSE
PINNs (JCP'19)	0.4101	0.4141	0.9803	0.9785	0.8514	0.8989	0.8956	0.9404
QRes (ICDM'21)	0.5349	0.5265	0.9826	0.9830	0.9035	0.9245	0.8381	0.8800
FLS (TAI'22)	0.1020	0.1190	0.0220	0.0390	0.1730	0.1970	0.7491	0.7866
PINNsFormer (ICLR'24)	0.3559	0.3632	0.0146	0.0296	0.4527	0.5217	0.2129	0.2236
RoPINNs (NeurIPS'24)	0.1650	0.1720	0.0070	0.0170	0.6350	0.7200	0.1545	0.1622
KAN (ICLR'25)	0.1433	0.1458	0.0166	0.0343	0.6049	0.6587	0.0901	0.1042
PINNMamba (ICML'25)	0.0197	0.0199	0.0094	0.0217	0.0188	0.0201	0.0535	0.0583
PIANO (ours)	**0.0057**	**0.0059**	**0.0001**	**0.0008**	**0.0032**	**0.0104**	**0.0000**	**0.0002**
Promotion (%)	71.1	70.4	98.6	95.3	83.0	48.3	100.0	99.7

The physics-informed component governs the optimization process. Instead of comparing the predicted rollout to a ground-truth solution, the loss function measures how well the generated trajectory satisfies the governing physical laws. This is done by evaluating the residuals of the underlying partial differential equation (PDE), as well as the errors in satisfying the boundary conditions. These residuals are computed over the full predicted spatiotemporal grid, using finite difference approximations for both spatial and temporal derivatives. The total loss is then aggregated over all points and time steps in the trajectory.

The model parameters are updated through backpropagation through time (BPTT). By maintaining gradient flow through the full autoregressive sequence, the model learns a stable state transition function, or evolution operator, that captures long-range temporal dependencies and adheres to the physical constraints. This end-to-end training on physically constrained rollouts directly minimizes the one-step rollout error discussed in Theorem 3.4. As a result, the model mitigates error accumulation commonly found in non-autoregressive approaches and produces stable, accurate long-term predictions.

C PDE Benchmark Experiment

Benchmarks We evaluate PIANO on four time-dependent PDE benchmarks: the Wave, Reaction, Convection, and Heat equations. These benchmarks are widely used in the literature [Zhao et al., 2024, Xu et al., 2025] and span diverse numerical challenges: higher-order derivatives (Wave), nonlinear dynamics (Reaction), numerical stiffness (Heat), and transport-dominated behavior prone to numerical diffusion (Convection).

Baselines We benchmark PIANO against a broad set of baselines: classical PINN variants (MLP-based PINNs [Raissi et al., 2019], First-Layer Sine networks (FLS) [Wong et al., 2022], and Quadratic Residual Networks (QRes) [Bu and Karpatne, 2021]); recent advances (Kolmogorov–Arnold Networks (KANs) [Liu et al., 2025] and Region-Optimized PINNs (RoPINNs) [Wu et al., 2024]); and state-of-the-art sequential models (PINNsFormer [Zhao et al., 2024] and PINNMamba [Xu et al., 2025]), which are sequential but not autoregressive—ideal for testing PIANO's autoregressive advantage.

Implementation Details PIANO is implemented as a state-space architecture and trained on a 200×200 discretized spatio-temporal grid using the AdamW optimizer. All models are trained on approx. same number of samples. For fairness, baselines rely on official implementations with reported hyperparameters and training routines. Regarding computational cost, PIANO is comparable to sequential baselines like PINNsFormer and PINNMamba, while its simpler state-space architecture offers efficiency gains. Performance is measured using relative Mean Absolute Error (rMAE) and

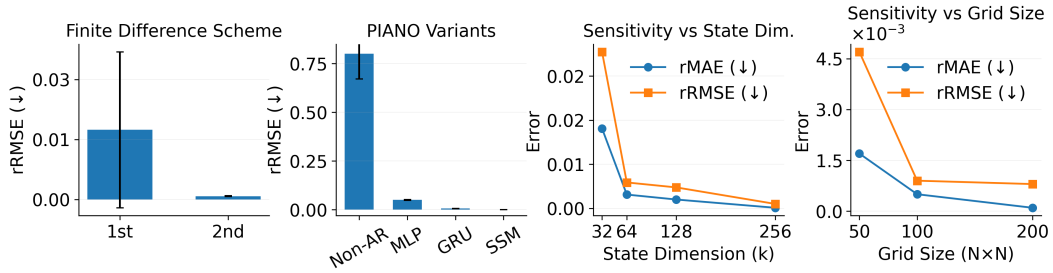


Figure 5: Ablations and hyperparameter guidance. The plots show the effect of finite difference order, the performance of different PIANO variants, and sensitivity to state dimension and training grid resolution. Error bars denote standard deviations across ten runs.

relative Root Mean Squared Error (rRMSE), which are standard metrics in the PINN literature [Xu et al., 2025].

Results Table 1 summarizes the PDE benchmark results. Pointwise PINNs suffer from high errors (consistent with Theorem A.3), while sequential models such as PINNMamba perform better. PIANO, however, consistently sets a new state of the art across all four benchmarks. For the Reaction and Heat equations, errors are driven to near zero with about 100% promotion over the second best model, while on the more challenging Wave and Convection equations PIANO outperforms PINNMamba by 70–80% promotion. These results highlight the accuracy and robustness of our autoregressive formulation for time-dependent PDEs.

Ablations and Hyperparameter Guidance. Figure 5 summarizes the ablation and hyperparameter guidance for PIANO. The experiments are conducted on the Reaction equation. The ablation studies demonstrate that a second-order finite difference scheme yields substantially lower error than the first-order version, confirming the importance of accurate derivative approximations. They also show that autoregression is critical: while a non-autoregressive baseline performs poorly, progressively richer backbones (MLP, GRU, SSM) with PIANO deliver significant gains, with the SSM achieving the lowest error. The sensitivity analysis further highlights that increasing the state dimension and training grid resolution consistently reduces error, with diminishing returns once $k = 256$ and a 200×200 grid are reached. Together, these findings validate the design of PIANO and provide practical guidance for hyperparameter choices in PDE learning.

D Weather Forecasting

D.1 Setup and Implementation

Background Weather forecasting has traditionally been dominated by numerical simulations of complex atmospheric physics. Although powerful, these methods are computationally demanding. Recently, deep learning models have emerged as a promising alternative, yet they often function as a “black-box" that neglect the underlying physical principles. A more robust approach involves integrating physical laws with deep learning approaches. ClimODE [Verma et al., 2024] is a recent model that successfully applies the physics-informed strategy to weather forecasting. It is built on a core principle from statistical mechanics: weather evolution can be described as a continuous-time advection process, which models the spatial movement and redistribution of quantities like temperature and pressure. By framing the problem as a neural Ordinary Differential Equation (ODE) that adheres to the advection equation, ClimODE enforces value-conserving dynamics, a strong inductive bias that leads to more stable and physically plausible forecasts. With PIANO, we build on the ClimODE framework by introducing an autoregressive training scheme to further enhance predictive accuracy.

Setup Weather forecasting involves predicting the evolution of key atmospheric variables such as atmospheric temperature (`t`), surface temperature (`t2m`), horizontal wind components (`u10`, `v10`), and geopotential (`z`). We adopt the physics-informed framework of ClimODE [Verma et al., 2024],

which models weather evolution as a continuous-time process governed by a system of neural ODEs. This system jointly evolves the weather state, denoted by $u(t)$, and a corresponding velocity field, $v(t)$.

The ODE system has two components. The first governs the rate of change of the weather state, $\dot{u}$, and is constrained by the physical advection equation, which ensures that quantities are transported and conserved according to physical principles. The second component governs the rate of change of the velocity field, $\dot{v}$, which is learned by a neural network, f_θ. This network takes as input the current state $u(\tau)$, its spatial gradient $\nabla u(\tau)$, the velocity $v(\tau)$, and spatiotemporal embeddings ψ to determine the acceleration of the flow.

In PIANO, we use this same physics-informed ODE structure but introduce an autoregressive training strategy with teacher forcing to reduce error accumulation. Instead of forecasting the entire trajectory in one step, the model is conditioned on the ground truth from the previous time point. The revised forecast equation for a single time step from t_i to t_j is given by:

$$\begin{bmatrix} \hat{u}(t_j) \\ \hat{v}(t_j) \end{bmatrix} = \begin{bmatrix} y_i \\ v(t_i) \end{bmatrix} + \int_{t_i}^{t_j} \begin{bmatrix} -\nabla \cdot (\hat{u}(\tau)\hat{v}(\tau)) \\ f_\theta(\hat{u}(\tau), \nabla\hat{u}(\tau), \hat{v}(\tau), \psi) \end{bmatrix} d\tau,$$

where y_i denotes the observed ground truth state at time t_i, $v(t_i)$ is the inferred velocity at that time, and $\nabla\cdot$ is the spatial divergence operator.

We evaluate PIANO on the ERA5 dataset Rasp et al. [2020], a benchmark for global weather forecasting providing 6-hourly reanalysis data at 5.625° resolution for five variables: `t`, `t2m`, `u10`, `v10`, and `z`. We compare against several state-of-the-art baselines including Neural ODE (NODE) Verma et al. [2024], FCN Pathak et al. [2022], ClimaX Nguyen et al. [2023], and the original ClimODE Verma et al. [2024]. As a reference, we also report results for the gold-standard Integrated Forecasting System (IFS) ECMWF [2023] which is one of the most advanced global physics simulation model and has high computational demands. Performance is evaluated using two standard metrics: root mean square error (RMSE) and anomaly correlation coefficient (ACC). RMSE quantifies the absolute prediction error, while ACC measures the correlation between predicted and observed anomalies, capturing the directional accuracy. Both metrics are latitude-weighted to reflect the spherical geometry of the Earth.

Implementation Our experimental framework directly mirrors the setup used by ClimODE to ensure a fair comparison. The primary task is forecasting future atmospheric states based on an initial state, with lead times ranging from 6 to 36 hours. The model is implemented in PyTorch. The underlying system of ODEs is solved using the Euler method with a time resolution of 1 hour, managed by the 'torchdiffeq' library. All experiments are conducted on a single NVIDIA A100 GPU with 40GB of memory. The model is trained for 300 epochs using a batch size of 8. The learning rate is managed by a Cosine Annealing scheduler.

Evaluation Metrics We assess model performance using two standard meteorological metrics: latitude-weighted Root Mean Squared Error (RMSE) and Anomaly Correlation Coefficient (ACC), computed after de-normalizing the predictions.

$$\text{RMSE} = \frac{1}{N}\sum_{t=1}^{N}\left[\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}\alpha(h)(y_{thw} - u_{thw})^2\right]^{1/2} \tag{14}$$

$$\text{ACC} = \frac{\sum_{t,h,w}\alpha(h)\tilde{y}_{thw}\tilde{u}_{thw}}{\sqrt{\sum_{t,h,w}\alpha(h)\tilde{y}_{thw}^2 \sum_{t,h,w}\alpha(h)\tilde{u}_{thw}^2}} \tag{15}$$

Here, y_{thw} and u_{thw} denote the ground truth and model prediction at time t, latitude index h, and longitude index w, respectively. The term $\alpha(h) = \cos(h) \Big/ \frac{1}{H}\sum_{h'}\cos(h')$ represents the normalized latitude weight, accounting for the area distortion in latitude-longitude grids due to Earth's curvature.

The anomalies are computed by subtracting the empirical mean:

$$\tilde{y}_{thw} = y_{thw} - C, \quad \tilde{u}_{thw} = u_{thw} - C,$$

where $C = \frac{1}{N}\sum_t y_{thw}$.

ACC measures the correlation between predicted and true anomalies. Higher ACC indicates better skill in capturing deviations from climatological means. Latitude-weighted RMSE evaluates the spatial accuracy of forecasts while correcting for latitudinal area distortion. Lower RMSE and higher ACC both indicate better forecasting performance.

Dataset and Preprocessing We use the ERA5 dataset, as preprocessed for the WeatherBench benchmark. The data is provided at a 5.625° spatial resolution with a 6-hour time increment. Our experiments focus on five key variables: 2-meter temperature (`t2m`), temperature at 850 hPa (`t`), geopotential at 500 hPa (`z`), and the 10-meter U and V wind components (`u10`, `v10`). All variables are normalized to a [0, 1] range using min-max scaling. The dataset is partitioned by year, with 2006-2015 used for training, 2016 for validation, and 2017-2018 for testing.

D.2 Results

We evaluate PIANO's ability to forecast global weather variables using the ERA5 dataset. Figure 6 provides a visual analysis of PIANO's probabilistic predictions (extended from the ClimODE framework) at a fixed forecast time (2017-01-01T12:00) across five key atmospheric variables: geopotential height at 500 hPa (`z`), temperature at 850 hPa (`t`), 2-meter surface temperature (`t2m`), and the 10-meter U and V wind components (`u10`, `v10`). Each row corresponds to a variable, while the columns show the predicted mean (μ), upper bound ($\mu + \sigma$), predicted standard deviation (σ), and pointwise error. These results demonstrate that PIANO not only captures the spatial structure of each variable, but also quantifies predictive uncertainty effectively, with visually low error and consistent uncertainty estimates across regions.

All uncertainty visualizations (e.g., predicted σ maps) and uncertainty-based metrics (e.g., CRPS) shown in this paper are taken from the ClimODE's probabilistic emission model, specifically, the Gaussian observation model

$$y_k(x,t) \sim \mathcal{N}\big(u_k(x,t) + \mu_k(x,t),\, \sigma_k^2(x,t)\big),$$

with learnable bias μ_k and variance σ_k^2 trained via negative log-likelihood. As we extend ClimODE with PIANO the probabilistic emission model is naturally extended.

Quantitative results are summarized in Table 2, where we report latitude-weighted RMSE and Anomaly Correlation Coefficient (ACC) at multiple forecast lead times, comparing PIANO against several strong neural and numerical baselines, including NODE, ClimaX, FCN, IFS, and ClimODE. Across all variables and lead times, PIANO achieves state-of-the-art performance, often outperforming neural baselines by a significant margin and in some cases approaching the accuracy of IFS. The model shows particularly strong gains in mid-range horizons (12–24 hours), maintaining high correlation and low error while producing calibrated uncertainty estimates. These results highlight the benefit of PIANO's autoregressive, physics-informed structure for long-range, high-resolution weather modeling.

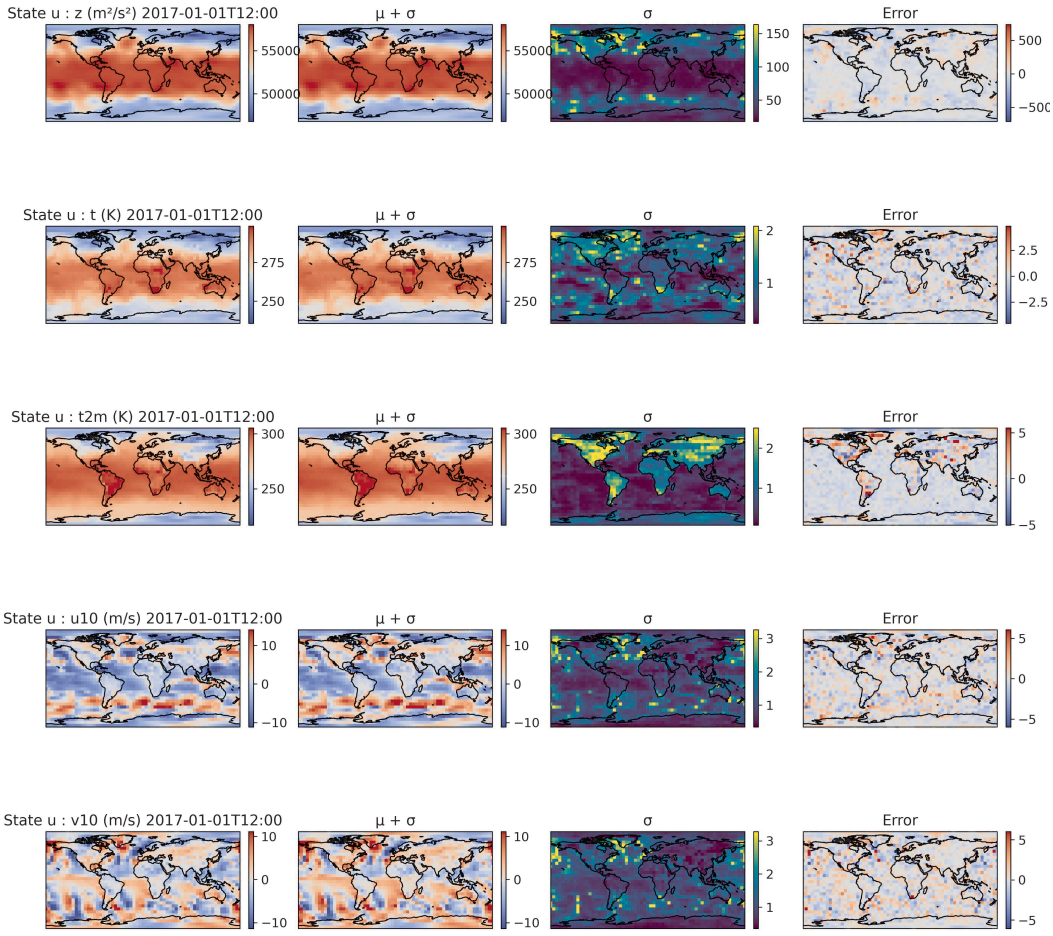


Figure 6: Visualization of PIANO's forecasting capabilities for five atmospheric state variables on 2017-01-01 at 12:00. Each row corresponds to a different variable: geopotential (`z`), atmospheric temperature (`t`), 2-meter surface temperature (`t2m`), and the 10-meter U-wind (`u10`) and V-wind (`v10`) components. The columns show (from left to right): the predicted mean μ, upper bound $\mu + \sigma$, predicted standard deviation σ, and prediction error (difference from ground truth). PIANO captures spatial structure and uncertainty across variables, with low errors. The errors are more pronounced where PIANO already suggests uncertainty (σ).

Table 2: Latitude weighted RMSE(↓) and Anomaly Correlation Coefficient (ACC(↑)) comparison with baselines on global forecasting on the ERA5 dataset. PIANO generally outperforms all the neural baselines.

		RMSE(↓)						ACC(↑)					
Variable	**Lead-Time (hrs)**	**NODE**	**ClimaX**	**FCN**	**IFS**	**ClimODE**	**PIANO (Ours)**	**NODE**	**ClimaX**	**FCN**	**IFS**	**ClimODE**	**PIANO (Ours)**
	6	300.64	247.5	149.4	26.9	102.9 ± 9.3	69.07 ± 4.99	0.96	0.97	0.99	1.00	0.99	1.00
	12	460.23	265.4	217.8	(N/A)	134.8 ± 12.3	109.07 ± 8.30	0.88	0.96	0.99	(N/A)	0.99	0.99
z	18	627.65	319.8	275.0	(N/A)	162.7 ± 14.4	145.99 ± 11.95	0.79	0.95	0.99	(N/A)	0.98	0.99
	24	877.82	364.9	333.0	51.0	193.4 ± 16.3	185.22 ± 15.91	0.70	0.93	0.99	1.00	0.98	0.98
	36	1028.20	455.0	449.0	(N/A)	259.6 ± 22.3	263.44 ± 22.96	0.55	0.89	0.99	(N/A)	0.96	0.97
	6	1.82	1.64	1.18	0.69	1.16 ± 0.06	0.92 ± 0.04	0.94	0.94	0.99	0.99	0.97	0.98
	12	2.32	1.77	1.47	(N/A)	1.32 ± 0.13	1.16 ± 0.05	0.85	0.93	0.99	(N/A)	0.96	0.97
t	18	2.93	1.93	1.65	(N/A)	1.47 ± 0.16	1.32 ± 0.06	0.77	0.92	0.99	(N/A)	0.96	0.96
	24	3.35	2.17	1.83	0.87	1.55 ± 0.18	1.48 ± 0.07	0.72	0.90	0.99	0.99	0.95	0.96
	36	4.13	2.49	2.21	(N/A)	1.75 ± 0.26	1.76 ± 0.09	0.58	0.86	0.99	(N/A)	0.94	0.94
	6	2.72	2.02	1.28	0.97	1.21 ± 0.09	1.01 ± 0.05	0.82	0.92	0.99	0.99	0.97	0.98
	12	3.16	2.26	1.48	(N/A)	1.45 ± 0.10	1.20 ± 0.09	0.68	0.90	0.99	(N/A)	0.96	0.97
t2m	18	3.45	2.45	1.61	(N/A)	1.43 ± 0.09	1.29 ± 0.08	0.69	0.88	0.99	(N/A)	0.96	0.97
	24	3.86	2.37	1.68	1.02	1.40 ± 0.09	1.42 ± 0.10	0.79	0.89	0.99	0.99	0.96	0.96
	36	4.17	2.87	1.90	(N/A)	1.70 ± 0.15	1.68 ± 0.15	0.49	0.83	0.99	(N/A)	0.94	0.94
	6	2.30	1.58	1.47	0.80	1.41 ± 0.07	1.24 ± 0.06	0.85	0.92	0.95	0.98	0.91	0.95
	12	3.13	1.96	1.89	(N/A)	1.81 ± 0.09	1.53 ± 0.07	0.70	0.88	0.93	(N/A)	0.89	0.93
u10	18	3.41	2.24	2.05	(N/A)	1.97 ± 0.11	1.74 ± 0.07	0.58	0.84	0.91	(N/A)	0.88	0.91
	24	4.10	2.49	2.33	1.11	2.01 ± 0.10	1.96 ± 0.09	0.50	0.80	0.89	0.97	0.87	0.88
	36	4.68	2.98	2.87	(N/A)	2.25 ± 0.18	2.35 ± 0.12	0.35	0.69	0.85	(N/A)	0.83	0.83
	6	2.58	1.60	1.54	0.94	1.53 ± 0.08	1.30 ± 0.06	0.81	0.92	0.94	0.98	0.92	0.95
	12	3.19	1.97	1.81	(N/A)	1.81 ± 0.12	1.58 ± 0.07	0.61	0.88	0.91	(N/A)	0.89	0.92
v10	18	3.58	2.26	2.11	(N/A)	1.96 ± 0.16	1.79 ± 0.08	0.46	0.83	0.86	(N/A)	0.88	0.90
	24	4.07	2.48	2.39	1.33	2.04 ± 0.10	2.01 ± 0.09	0.35	0.80	0.83	0.97	0.86	0.88
	36	4.52	2.98	2.95	(N/A)	2.29 ± 0.24	2.40 ± 0.13	0.29	0.69	0.75	(N/A)	0.83	0.82